Indian Institute Of Technology Madras

ID5130 Parallel Scientific Computing

Assignment 1

AE20B027 Hareesh P Nair

Question 1

Solve the following system of linear equations using recursive-doubling algorithm by clearly showing all the required steps. You may use the recursive expressions that we have developed or solve it directly using the philosophy of the method

$$3x_1 - x_2 = 2$$
$$-x_1 + 3x_2 - x_3 = 1$$
$$-x_2 + 3x_3 - x_4 = 1$$
$$-x_3 + 3x_4 = 2$$

Solution

Consider the given equations be labelled as:

$$3x_1 - x_2 = 2 \tag{1}$$

$$-x_1 + 3x_2 - x_3 = 1 \tag{2}$$

$$-x_2 + 3x_3 - x_4 = 1 \tag{3}$$

$$-x_3 + 3x_4 = 2 \tag{4}$$

Consider equation (1) and (2), from equation (2), we get

$$x_2 = \frac{1 + x_1 + x_3}{3}$$

Substituting value of x_2 in equation (1), we get

$$3x_1 - \left(\frac{1 + x_1 + x_3}{3}\right) = 2$$

Multiplying the above equation by 3, we get

$$8x_1 - x_3 = 7 \tag{5}$$

Now consider equation (1), (2) and (3),

$$x_1 = \frac{2 + x_2}{3} \text{ from equation (1)}$$

$$x_3 = \frac{1 + x_2 + x_4}{3} \text{ from equation (3)}$$

Substituting them in equation (2) and multiplying by 3, we get

$$7x_2 - x_4 = 6 \tag{6}$$

Now consider equation (2), (3) and (4),

$$x_2 = \frac{1 + x_3 + x_4}{3} \text{ from equation (2)}$$

$$x_4 = \frac{2 + x_3}{3} \text{ from equation (4)}$$

Substituting them in equation (2) and multiplying by 3, we get

$$7x_3 - x_1 = 6 \tag{7}$$

Consider equation (3) and (4)

$$x_3 = \frac{1 + x_2 + x_4}{3} \text{ from equation (3)}$$

substituting in equation (4) and multiplying by 3, we get

$$8x_4 - x_2 = 7 \tag{8}$$

Now consider equation (5) and (7),

$$x_3 = 8x_1 - 7$$

Substituting in equation (7), we get $x_1 = 1$
Similarly considering equation (6) and (8), we get $x_2 = 1$
Similarly considering equation (5) and (7) again, we get $x_3 = 1$
Lastly, considering equation (6) and (8) again, we get $x_4 = 1$
Hence,

$$\boxed{x_1 = 1} \quad \boxed{x_2 = 1} \quad \boxed{x_3 = 1} \quad \boxed{x_4 = 1}$$

Question 2

Consider the calculation of the derivative of the following function,

$$f(x) = \sin(5x) \quad 0 \leq x \leq 3$$

using four-order accurate Padé scheme for the interior and third-order accurate one-sided Padé scheme near the boundaries, given as follows,

$$f'_{j+1} + 4f'_j + f'_{j-1} = \frac{3}{h}\left(f_{j+1} - f_{j-1}\right)$$

where j is any interior point ($j = 1, 2, ..., n-1$) and

$$f'_0 + 2f'_1 = \frac{1}{h}\left(-\frac{5}{2}f_0 + 2f_1 + \frac{1}{2}f_2\right)$$

$$f'_n + 2f'_{n-1} = \frac{1}{h}\left(-\frac{5}{2}f_n - 2f_{n-1} + \frac{1}{2}f_{n-2}\right)$$

where h is the grid spacing and n is the number of grid points in the x direction.

a Develop a serial program to compute the derivative using the tridiagonal LU decomposition. Plot the analytical solution and the numerical solution obtained for $n = 25$.

b Develop an OpenMP program to compute the derivative using the recursive-doubling algorithm. Plot the analytical solution and the numerical solution for $n = 100$ and number of threads $p = 2$. Plot the time-taken by the solver for $n = 1000$ for number of threads $p = 2, 4$ and 8.

Solution - a

The serial code has been developed in C++ and uploaded to the relevant submission link. The plot for the four-order Padé scheme is obtained from the code is given below

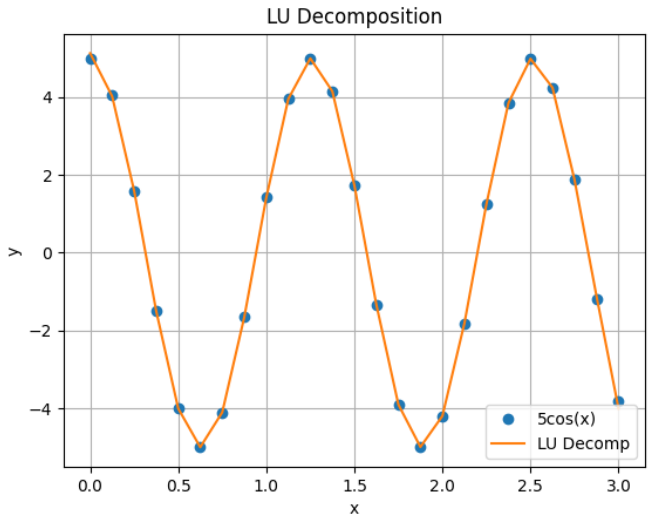

Figure 1: Plot of four-order Padé scheme solution using LU Decomposition

Solution - b

The code for the Recursive Doubling method for solving the Padé scheme has been developed and uploaded. The plot for the Recursive Doubling solution for $n = 100$ is given below:

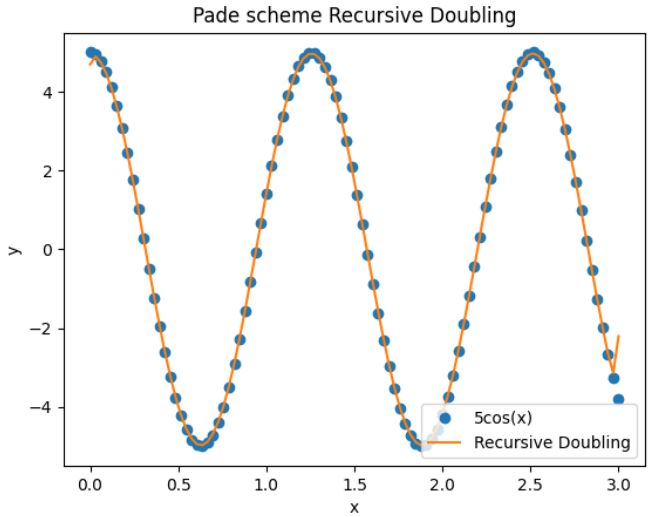

Figure 2: Plot of four-order Padé scheme solution using Recursive Doubling

The plot of time taken for different number of threads $p = 2, 4$ and 8 for solving the Recursive Doubling algorithm is given below:

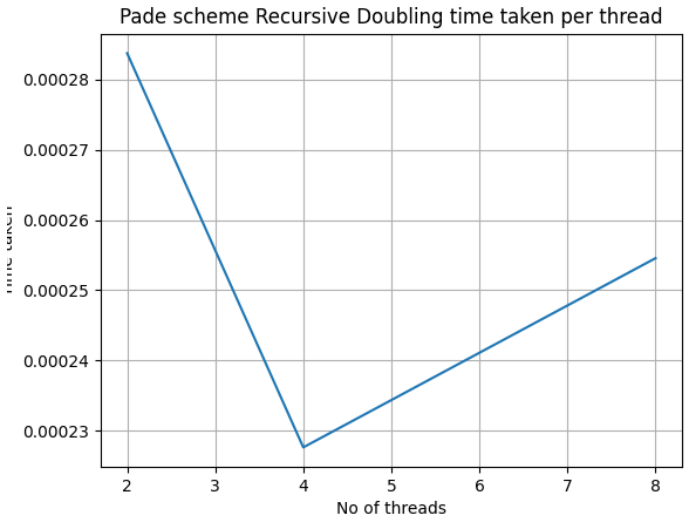

Figure 3: Time taken for different number of threads to solve Recursive Doubling algorithm

In the above plot, we can see that for $p = 8$, the time taken is more than for $p = 4$. This can be explained by the overhead time taken by 8 threads to be initialized is itself more than the calculation of the algorithm by 8 threads. Hence, we get this trend. In cases where the number of points used of discretization, n, is much higher, we can avoid such scenarios.

Question 3

Consider the solution of the following Poisson equation,

$$\nabla^2\phi = -q; \quad q = 2(2 - x^2 - y^2); \quad \phi(\pm 1, y) = 0; \quad \phi(x, \pm 1) = 0$$

using Gauss-Seidel method. The discretized equation using this method can be written as follows,

$$\phi_{i,j}^{(k+1)} = \frac{1}{4}\left[\phi_{i+1,j}^{(k)} + \phi_{i-1,j}^{(k+1)} + \phi_{i,j+1}^{(k)} + \phi_{i,j-1}^{(k+1)}\right] + \frac{\Delta^2}{4} q_{i,j}$$

where $\Delta = \Delta x = \Delta y$. Consider an initial guess of $\phi(x, y) = 0$ everywhere. The exact solution to this problem is given by $\phi = (x^2 - 1)(y^2 - 1)$. Use double precision arithmetic.

a Develop a serial Gauss-Seidel program to solve the discretized equations and solve the system using $\Delta = \Delta x = \Delta y = 0.1$ (that is 21 points along each of the directions). Report the number of iterations required to bring the numerical solution to within 1% of the exact solution. Plot the numerical and the analytical solutions of ϕ vs x for $y = 0.5$.

b Develop an OpenMP program for Gauss-Seidel method using the (i) diagonal approach (ii) red-black coloring approach.

c For each of the parallel programs developed above, verify that your parallel program is giving the same result as that of the serial program that you developed above for a grid of $\Delta = 0.1$ by plotting the two results on the same graph. Upon successful verification, run this program for $\Delta = 0.1,\ 0.01, \text{and}\ \ 0.005$ using 8 threads. Plot the time-taken by the parallel solvers and the serial solver as a function of the grid size (Δ). Do you see any improvement in performance? Which parallel method is better ?

d For a grid size of $\Delta = 0.005$, plot the time-taken by each of the parallel solvers as a function of the number of threads. Consider the number of threads to be $p = 2, 4, 8$ and 16. Which parallel method is better?

Solution - a

The serial code for Gauss Seidel has been developed in C++ and the following plot was obtained for the Gauss Seidel serial scheme for $\Delta = 0.1$

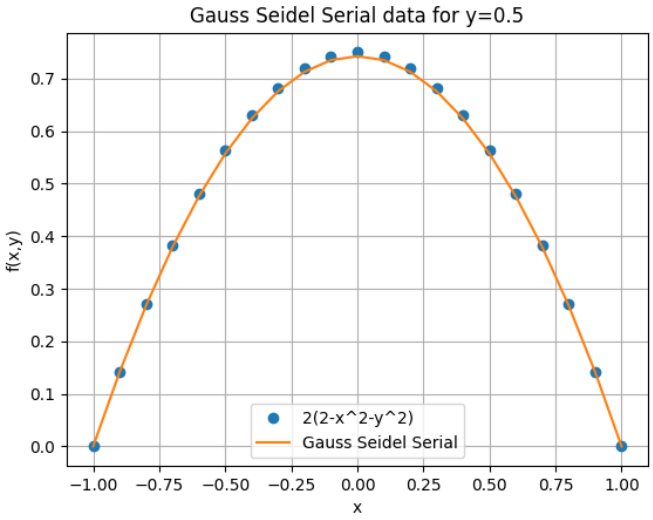

Figure 4: Plot of $\phi(x, y = 0.5)$ for Gauss Seidal scheme solution

Solution - b

The code for the Diagonal Wave method and Red Black coloring for solving the Gauss Seidel scheme has been developed using C++ and uploaded to the relevant submission link. The following set of answers will dive into more detail about them

Solution - c

The values of the Diagonal Wave method and Red-Black coloring method is closely matching with the serial solution for $\phi(x, y = 0.5)$. This can be visulaized in the below plot

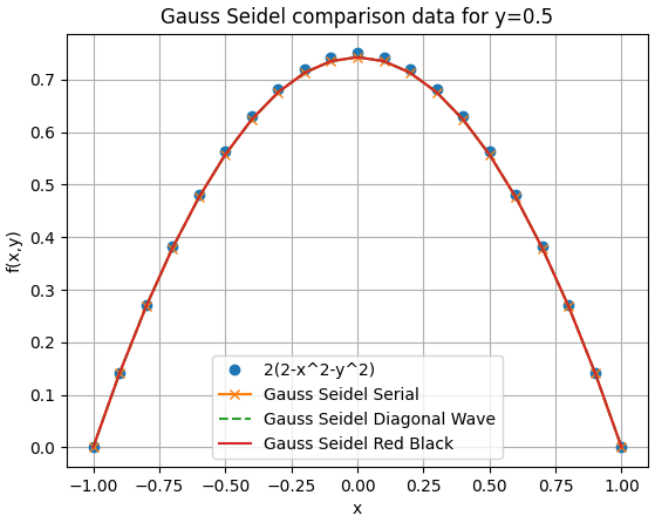

Figure 5: Plot of $\phi(x, y = 0.5)$ using different algorithm for Gauss Seidel scheme

Now that the result of the parallel programs has been verified, the time taken by each parallel program for solving the Poisson equation by following different algorithm on Gauss Seidel method for different grid sizes $\Delta = 0.1,\ 0.01,$ and 0.005 for $p = 8$ is recorded. The following plot can provide a visualisation on the time taken

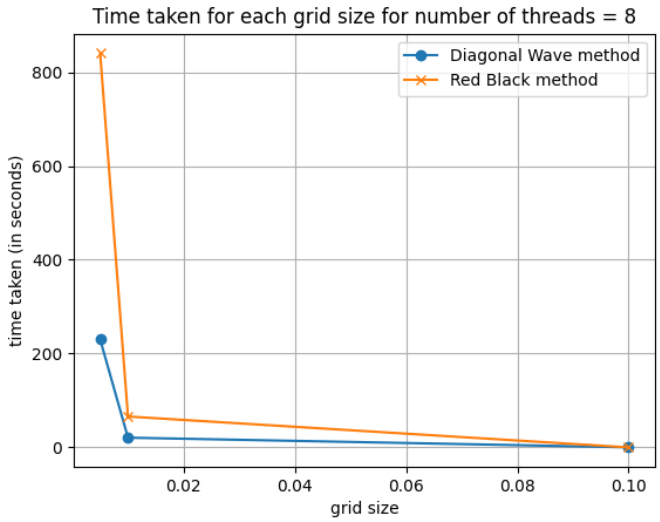

Figure 6: Time taken for 8 threads to perform Gauss Seidel on different grid sizes

As we can see from the above plot, the time taken by each algorithm is different for each grid sizes. From observation we can say **Diagonal Wave method performs much faster than Red Black coloring method** for $p = 8$ number of threads. Also the time taken for smaller grid size is much higher than the time taken for a larger grid size.

Solution - d

The following plot was obtained for the time taken for solving the Poisson equation by the Gauss Seidel iteration method using different algorithms for $\Delta = 0.005$ and $p = 2, 4, \text{and}, 8$. (Note: **I cannot use 16 threads as my laptop is hexacore**).

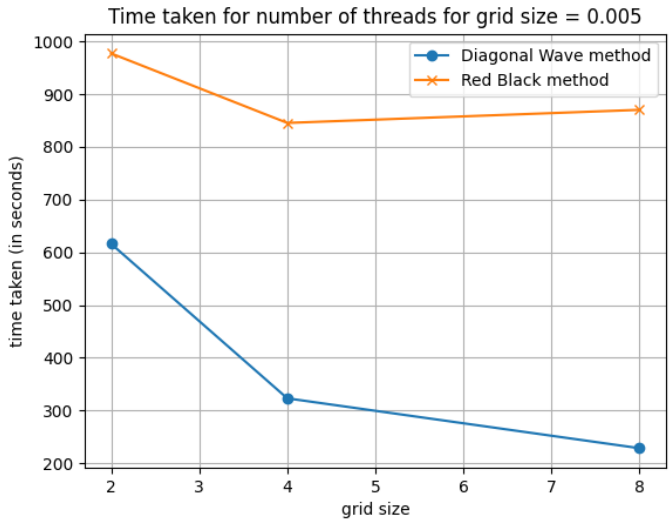

Figure 7: Time taken for different number of threads to perform Gauss Seidel with grid size = 0.005

From the plot, its obvious that **Diagonal Wave Method** is performing much faster compared to Red-Black-coloring scheme